This document includes some brief guidelines for how to adjust Geant4 physics processes in LArSoft. I learned most of this while trying to implement G4AntiNeutronAnnihilationAtRest, so some of these instructions may not hold true in general. In any case, I hope this is helpful!

If you are interested in using a specific Geant4 physics process in LArSoft, you should first determine whether the process is already included in the default physics list. The current version of LArSoft uses the QGSP_BERT physics list from Geant4 9.6.p02 (you should probably verify this, since there might have been changes made to the list in more recent versions).

If the process you want to study is not included in the default physics list, you will need to add the physics process (see **1**). If the process is included in the physics list but does not appear (or appears more infrequently that you would like), it might be helpful to adjust the energy range of the competing processes (see **2**). Note: The name of the Geant4 process that created a simb::MCParticle object can be accessed by doing:

```
myParticle.Process()
```

You can tell what processes Geant4 uses in your particular set of generated events by printing out the processes that create each particle.

1. Adding the physics process In order for a particle to undergo a particular physics process, the process must be registered with the particle's G4ProcessManager. Some brief examples are included in the Geant4 documentation, and I include a lengthier discussion of how I added AntiNeutronAnnihilationAtRest here.

The process managers for each particle can be accessed in the file larsim/LArG4/PhysicsList.cxx. Be sure to include the .hh file associated with the process that you want to add at the beginning of PhysicsList.cxx. In the loop over all particles, select a particular type of particle, then create the physics process and add it to the particle's process manager. For example, the following code was used to add AntiNeutronAnnihilationAtRest:

```
if(particle->GetParticleName() == G4AntiNeutron::AntiNeutron()->GetParticleName()){
     G4ProcessManager* pmanager = particle->GetProcessManager();
     pmanager = particle->GetProcessManager();
     G4AntiNeutronAnnihilationAtRest* theanAnnihilation =
          new G4AntiNeutronAnnihilationAtRest();
     pmanager->AddRestProcess(theanAnnihilation);
     pmanager->SetProcessActivation(theanAnnihilation,true);
}
```

The full code can be found at /uboone/app/users/jennetd/nnbar/srcs/larsim/LArG4/PhysicsList.cxx.

Alternatively, you can create the process first, then check whether it is applicable to each particle. If the process you are adding is valid for more than one type of particle, this may make more sense. Try something like:

```
G4AntiNeutronAnnihilationAtRest* theanAnnihilation =
     new G4AntiNeutronAnnihilationAtRest();
if(theanAnnihilation->IsApplicable(particle)){...}
```

If the process has been successfully added, it should appear under the list of available processes that is output when you run your event generator .fcl file (e.g. prodsingle.fcl). A sample of the output of prodsingle.fcl, with AntiNeutronAnnihilationAtRest added:

```
                                Hadronic Processes for <anti_neutron>
                                -----------------------------------
AntiNeutronInelastic  Models:     G4LEAntiNeutronInelastic: Emin(GeV)=    0  Emax(GeV)= 25
                                  G4HEAntiNeutronInelastic: Emin(GeV)=   20  Emax(GeV)= 100000

AntiNeutronInelastic  Crs sctns:             AntiAGlauber: Emin(GeV)=    0  Emax(GeV)= 1.79769e+305
                                          GheishaInelastic: Emin(GeV)=    0  Emax(GeV)= 100000

          hadElastic  Models:                hElasticLHEP: Emin(GeV)=    0  Emax(GeV)= 100000

          hadElastic  Crs sctns:           GheishaElastic: Emin(GeV)=    0  Emax(GeV)= 100000

          AntiNeutronAnnihilationAtRest
```

Note: if you are unsure whether a process is included in the default physics list, check and see if the event generator lists it here.

2. Adjusting the energy range for processes If the process that you are interested in does not occur in your generated events, it might be that another type of interaction is dominant in the energy range that you are dealing with. This did not turn out to be the case for AntiNeutronAnnhilationAtRest (which is only valid for antineutrons with $\vec{p} = 0$), but this strategy could prove useful for other studies.

If there are multiple processes that can occur in the energy range that you are studying, you can get the less interesting processes out of your way by telling them not to occur in the energy range that you are interested in. This allows rarer processes to show up more often.

For example, by default the low energy AntiNeutronInelastic process can occur between 0 and 25 GeV. This range is listed in the output of the event generator:

```
                                Hadronic Processes for <anti_neutron>
                                ---------------------------------
AntiNeutronInelastic  Models:     G4LEAntiNeutronInelastic: Emin(GeV)=    0  Emax(GeV)= 25
                                  G4HEAntiNeutronInelastic: Emin(GeV)=   20  Emax(GeV)= 100000

AntiNeutronInelastic  Crs sctns:            AntiAGlauber: Emin(GeV)=    0  Emax(GeV)= 1.79769e+305
                                          GheishaInelastic: Emin(GeV)=    0  Emax(GeV)= 100000

          hadElastic  Models:               hElasticLHEP: Emin(GeV)=    0  Emax(GeV)= 100000

          hadElastic  Crs sctns:          GheishaElastic: Emin(GeV)=    0  Emax(GeV)= 100000

          AntiNeutronAnnihilationAtRest
```

The following code will bump the minimum energy from 0 to 10 GeV:

```
if(particle->GetParticleName() == G4AntiNeutron::AntiNeutron()->GetParticleName()){
     G4ProcessManager* pmanager = particle->GetProcessManager();
     pmanager = particle->GetProcessManager();

     G4AntiNeutronInelasticProcess* theanInelasticProcess =
          new G4AntiNeutronInelasticProcess();
     G4LEAntiNeutronInelastic* theanLEPModel = new G4LEAntiNeutronInelastic();
     theanLEPModel->SetMinEnergy(10.0*GeV);
     theanInelasticProcess->RegisterMe(theanLEPModel);
     pmanager->AddDiscreteProcess(theanInelasticProcess);
}
```

The maximum energy can also be adjusted (see Geant4 documentation). Changes to the energy range of a process ought to be reflected in the output of the event generator when you recompile and run it again.